Increase the feedback in IoT development in Node-RED - *Quasi-Experiment*

Node-RED is one of the most widespread VPLs targeting IoT systems that mashups hardware devices, APIs, and third-party services, in a hybrid text-visual programming approach. With Node-RED, it is possible to create flows with the aim of composing some rules that the system has to comply with. These flows can be something like turning on the coffee-machine whenever the user wakes up, turning on the heating system when the user is returning home after work, turning on the A/C if the temperature rises above a certain level, and many other examples.

So, let's begin with a tutorial in order to become familiar with the tools that will be used in the next set of tasks.

Tutorial

Task 1

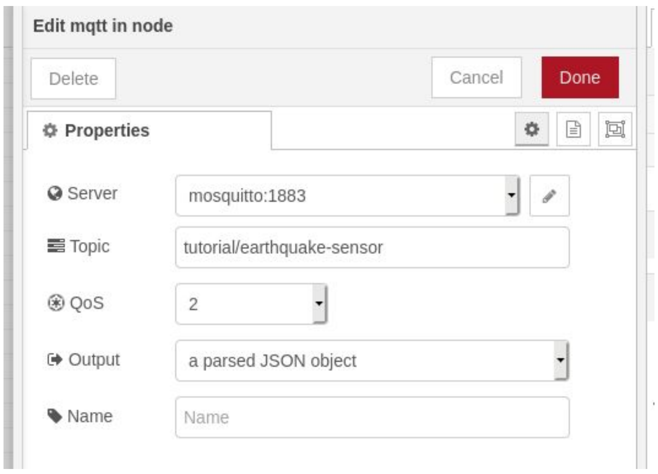
This task is similar to a "Hello World" where Node-RED is introduced alongside some basic concepts by creating a flow that demonstrates the Inject, Debug and Function nodes.

1. Go to http://nodered-feedback.com/nodered where the tool is running.
2. Go to https://nodered.org/docs/tutorials/first-flow and proceed with the tutorial skipping the first step.

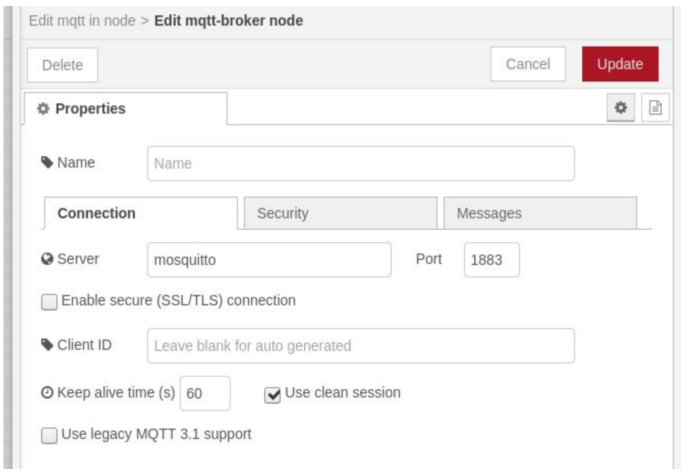
Task 2

This task will introduce a flow based on data from an earthquake sensor to do something useful. Also, it will introduce four new nodes (MQTT in, MQTT out, Switch and Change) that will be useful in further experiments.

1. Add an MQTT IN node, click on it to see further information and edit as follow:



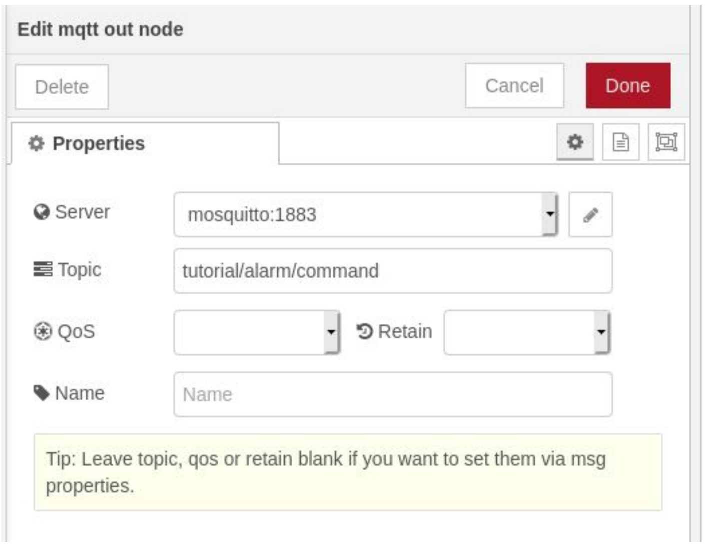
Server: mosquitto:1883
Topic: tutorial/earthquake-sensor
QoS: 2
Output: a parsed JSON object
Name: Name



Name: Name
Server: mosquitto
Port: 1883
☐ Enable secure (SSL/TLS) connection
Client ID: Leave blank for auto generated
Keep alive time (s): 60
☑ Use clean session
☐ Use legacy MQTT 3.1 support

More information about MQTT.

2. Add a Debug node to the output. Deploy the system and observe the data in the Debug Sidebar.
3. Add a Change node, wired to the output of the MQTT IN node. Configure it to set msg.payload to msg.payload.mag.
4. Add a Switch node to the workspace. Edit its properties and configure it to check the property msg.payload with a test of >= change it to test on a number and the value 7. Click Done to close and add a wire from the Change node node to this Switch node.
5. Add a Change node, wired to the output of the Switch node. Configure it to set msg.payload to the string ON.
6. Wire a new Debug node to the output of the Change node.
7. Wire a new MQTT out node to the output of the Change node with the following configuration:



Server: mosquitto:1883
Topic: tutorial/alarm/command
QoS:
Retain:
Name: Name

Tip: Leave topic, qos or retain blank if you want to set them via msg properties.

8. Deploy the flow to the runtime by clicking the Deploy button.

In the Debug sidebar you should see a list of entries with some contents that look like:

```
msg.payload : Object
{"time":"2017-11-19T15:09:03.120Z","latitude":-
21.5167,"longitude":168.5426,"depth":14.19,"mag":6.6,"magType":"mww","gap":21,"dmin":0.478,"rms":0
.86,"net":"us","id":"us2000brgk","updated":"2017-11-19T17:10:58.449Z","place":"68km E of Tadine,
New
Caledonia","type":"earthquake","horizontalError":6.2,"depthError":2.8,"magError":0.037,"magNst":72
,"status":"reviewed","locationSource":"us","magSource":"us"}
```

You can now click on the little arrow to the left of each property to expand them and examine the contents. Also, if there were any quakes with a magnitude greater than 7 you will also see debug messages like:

```
msg.payload : string(2)
"ON"
```

You could change the switch value of 7 to a smaller one to test your program. Remember to click on deploy after the change. ***NOTE:** This tutorial is based on the Second-Flow tutorial from Node-RED.

Feedback-nodered

This section describes the extra functionalities provided by the **feedback-nodered** tool. The messages shown in the node by default are the input messages. However, in nodes without any input, the output is shown. Also, these nodes without inputs are not able to inject messages or use breakpoint functionalities.

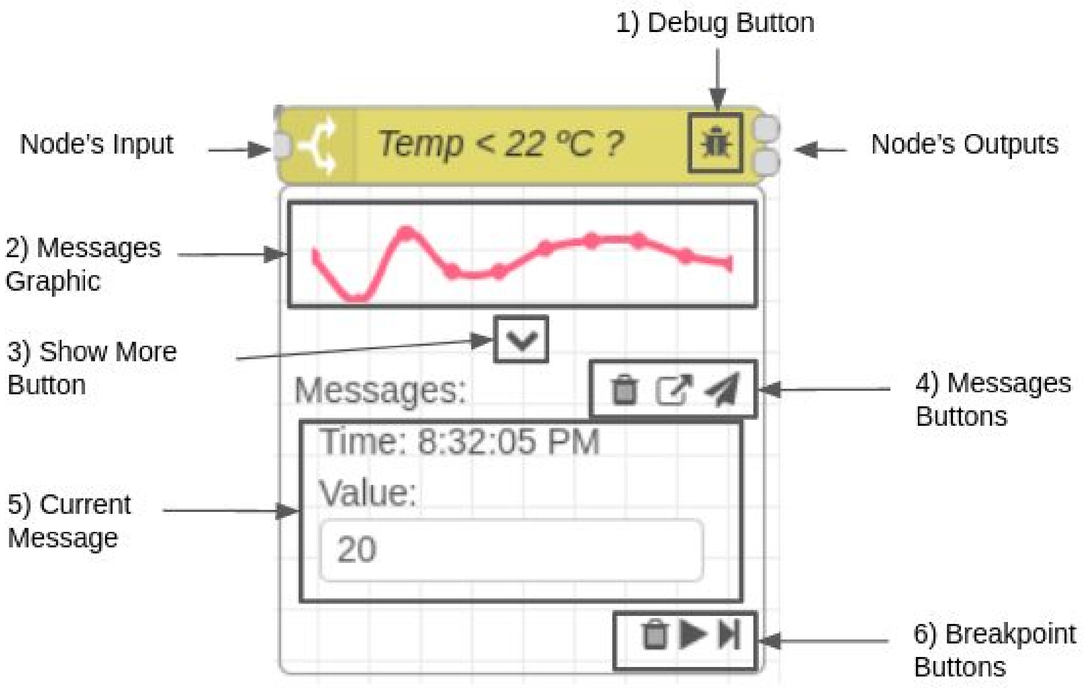


Figure 1.1

1) Debug Button - toggle the debug functionalities.

2) Messages Graphic - has two different modes, when a payload is a number it displays a line graph, as visible in figure 1.2, otherwise displays a scatter graph with message frequency, as shown in figure 1.3.

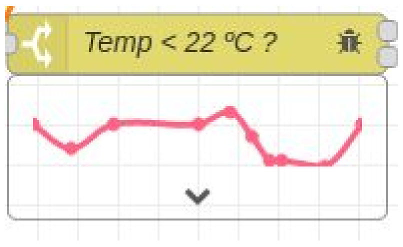


Figure 1.2

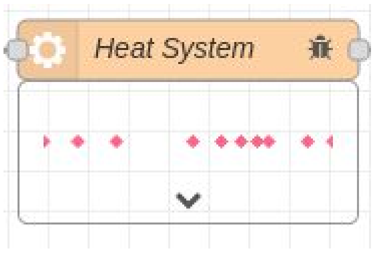
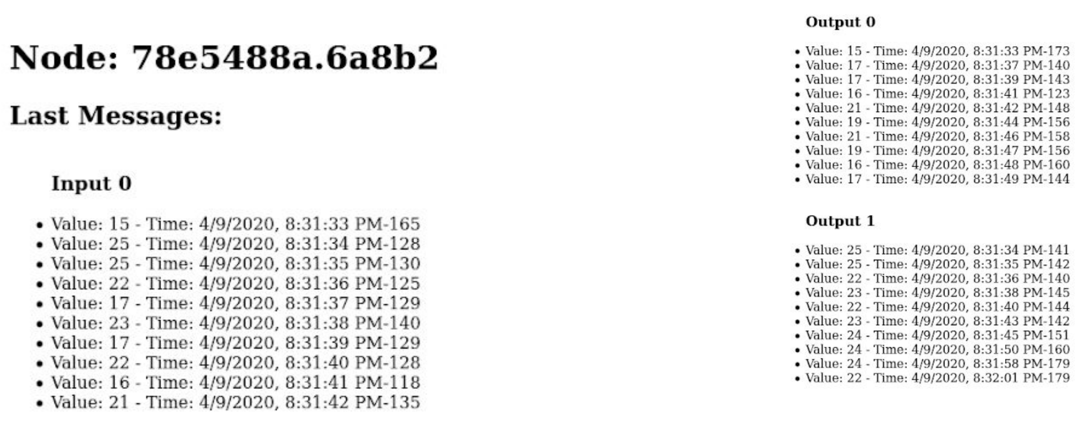


Figure 1.3

3) Show More Button - toggle all information above the button.

4) Messages Buttons:
    a) Trash Button - Clear all the current saved messages
    b) Expand Button - Show in a popup all the messages according to the input/output, as shown in figure 1.4:



Node: 78e5488a.6a8b2

Last Messages:

Input 0

- Value: 15 - Time: 4/9/2020, 8:31:33 PM-165
- Value: 25 - Time: 4/9/2020, 8:31:34 PM-128
- Value: 25 - Time: 4/9/2020, 8:31:35 PM-130
- Value: 22 - Time: 4/9/2020, 8:31:36 PM-125
- Value: 17 - Time: 4/9/2020, 8:31:37 PM-129
- Value: 23 - Time: 4/9/2020, 8:31:38 PM-140
- Value: 17 - Time: 4/9/2020, 8:31:39 PM-129
- Value: 22 - Time: 4/9/2020, 8:31:40 PM-128
- Value: 16 - Time: 4/9/2020, 8:31:41 PM-118
- Value: 21 - Time: 4/9/2020, 8:31:42 PM-135

Output 0

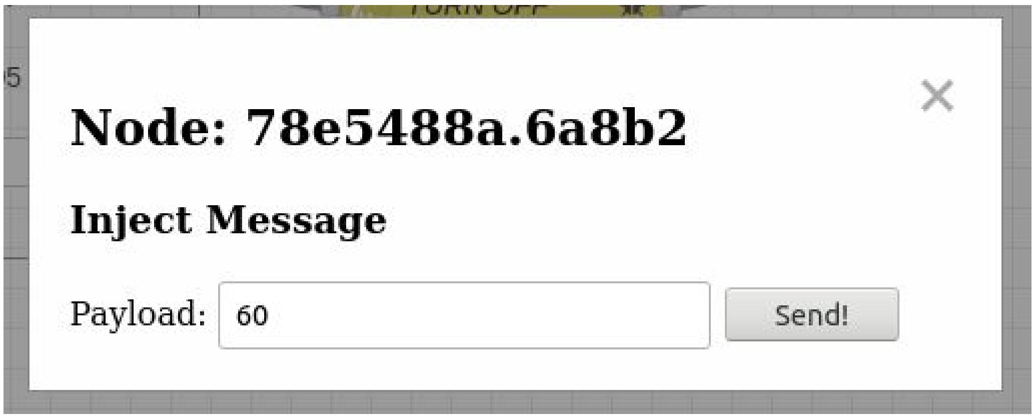
- Value: 15 - Time: 4/9/2020, 8:31:33 PM-173
- Value: 17 - Time: 4/9/2020, 8:31:37 PM-140
- Value: 17 - Time: 4/9/2020, 8:31:39 PM-143
- Value: 16 - Time: 4/9/2020, 8:31:41 PM-123
- Value: 21 - Time: 4/9/2020, 8:31:42 PM-148
- Value: 19 - Time: 4/9/2020, 8:31:44 PM-156
- Value: 21 - Time: 4/9/2020, 8:31:46 PM-158
- Value: 19 - Time: 4/9/2020, 8:31:47 PM-156
- Value: 16 - Time: 4/9/2020, 8:31:48 PM-160
- Value: 17 - Time: 4/9/2020, 8:31:49 PM-144

Output 1

- Value: 25 - Time: 4/9/2020, 8:31:34 PM-141
- Value: 25 - Time: 4/9/2020, 8:31:35 PM-142
- Value: 22 - Time: 4/9/2020, 8:31:36 PM-140
- Value: 23 - Time: 4/9/2020, 8:31:38 PM-145
- Value: 22 - Time: 4/9/2020, 8:31:40 PM-144
- Value: 23 - Time: 4/9/2020, 8:31:43 PM-142
- Value: 24 - Time: 4/9/2020, 8:31:45 PM-151
- Value: 24 - Time: 4/9/2020, 8:31:50 PM-160
- Value: 24 - Time: 4/9/2020, 8:31:58 PM-179
- Value: 22 - Time: 4/9/2020, 8:32:01 PM-179

Figure 1.4

c) Inject Message - Allow inject a message to debug purposes, as shown in Figure 1.5.



Node: 78e5488a.6a8b2

Inject Message

Payload: 60

Figure 1.5

5) Current Message:
    a) Time - Show the arrival time of a message
    b) Value - Show the payload of a message, however, if this payload is an object only displays [Object object]

6) Breakpoint Buttons:
    a) Trash Button - Clear queued messages when the breakpoint is activated
    b) Play/Pause Button - Allow stopping the messages in the current node, queuing them
    c) Step Button - Allow to send a message at a time and also editing them

Problem 1

John Doe has developed a system capable of automating the treatment of a strawberry plantation inside a greenhouse. However, strawberries require some special care such as:

- Exposure to sunlight;
- The soil pH must be contained between 5.5 to 7;
- Absence of wind;
- Temperature between 20-23 ºC;
- Humidity above 70%.

Currently, the system is capable of:

- To keep the **soil moisture** close to **75%**. Through a **humidity-temperature sensor** that communicates its values periodically and **if** they are **lower** than expected, the **irrigation system** is switched **ON**, **otherwise OFF**;

- To maintain the **temperature** through the **heating system**. This system is switched **ON** if the temperature is **below 20 ºC until** it reaches **23ºC**. Also, when **it's ON**, it automatically **closes** the **roof window**.

Note: You can access the **actuators status** in: http://localhost/simulator/agriculture/actuators . Also, In each task, there are tables that present information about the actuators and sensors that can be used or are already used in the system.

Task 1

John Doe has had huge headaches with the **heating system** that has not worked properly. Also, the **water system** isn’t working at all.

a) Open http://nodered-feedback.com/node-red/feedback
b) Identify and fix bugs.

Actuators

Name	Topic	Command to send
Roof Window	agriculture/roof-window/command	“1”/”0”
Irrigation System	agriculture/water-system/command	“ON”/”OFF”
Heating System	agriculture/heat-system/command	“ON”/”OFF”

Sensors

Name	Topic	Payload
Humidity-Temperature Sensor	agriculture/temp-hum-readings	{ "node-id":"sensor-node-3", "sensor": "dht11", "hum-percent": 60, "temp-C": 18, "timestamp": 1585673989 }

Task 2

Strawberries need to have **sunlight** daily. Implement a feature that, based on **weather forecasts**, turns **OFF** the **UV lamps** when it is a **sunny day** (you should use icon property from forecasting data with the value clear-day) . Otherwise, turn **ON** the **UV lamps** and **close** the **roof window**.

Actuators

Name	Topic	Command to send
Roof Window	agriculture/roof-window/command	“1”/”0”
Roof UV Lamps Controller	agriculture/roof-lux-controller/command	“ON”/”OFF”

Sensors

Name	Topic	Payload	Note

Forecasting	agriculture/forecasting	{
		"latitude": 41.1611,
		"longitude": -8.614,
		"timezone": "Europe/Lisbon",
		"time": 1586554471,
		"icon": "clear-day",
		"precipIntensity": 0,
		"temperature": 18.1,
		"dewPoint": 12.13,
		"humidity": 0.60,
		"pressure": 1022.4,
		"windSpeed": 1.32,
		"ozone": 306.3
		}
			Icon property will have one of the following values: clear-day, clear-night, rain, snow, sleet, wind, fog, cloudy, partly-cloudy-day, or partly-cloudy-night.

Problem 2

Now implement a system for a smart-home with some basic features:

- **Every day** at **6am**, turn **ON** the **water heater** and the **coffee machine**.
- **If** there is **movement** in the **kitchen**, turn **ON** the **lights**.

Note: You can access the **actuators status** in: http://localhost/simulator/smart-home/actuators . Also, in the tables above, there are information about the actuators and sensors that can be used.

Actuators

Name	Topic	Command to send
Water Heater Controller	smart-home/water-heat-controller/command	“ON”/”OFF”
Kitchen Lights	smart-home/kitchen-lights/command	“ON”/”OFF”
Coffee Machine	smart-home/coffee-machine/command	“ON”/”OFF”

Sensors

Name	Topic	Payload
Kitchen Motion Sensor	smart-home/kitchen-motion-readings	{ "node-id":"sensor-node-3", "sensor": "pir-motion", "motion-bool": 0, "timestamp": 1585673980 } **Note: motion-bool** property will have one of the following values: 0 if there is no movement, 1 otherwise.

Useful Resources

- Node-RED Documentation
- Node-RED Tutorials
- Node-RED Core Nodes Guide

Curiosity

Q: How to not overload a certain device? By overload, it means receiving messages at a higher rate.
A1: Using a Delay node configured to rate-limit the messages passing through it (Slow down messages passing through a flow).
A2: Using the same node but configured to rate-limit the messages passing through it with the option to drop intermediate messages enabled (Handle messages at a regular rate).

Q: How to not overload the device by constantly sending the same command?
A: Using a RBE node configured to send a message only when the value changed (Drop messages that have not changed value).

Q: How to not incessantly turn on/off a device when receiving slightly differences in data readings?
A1: Using a RBE node configured to only send a message if respect a certain threshold (e.g., check if our input data has changed by more than 20%).

Q: What can I do to automate my home?
A: 25 Home Automation Ideas: Ultimate Smart Home Tour!.